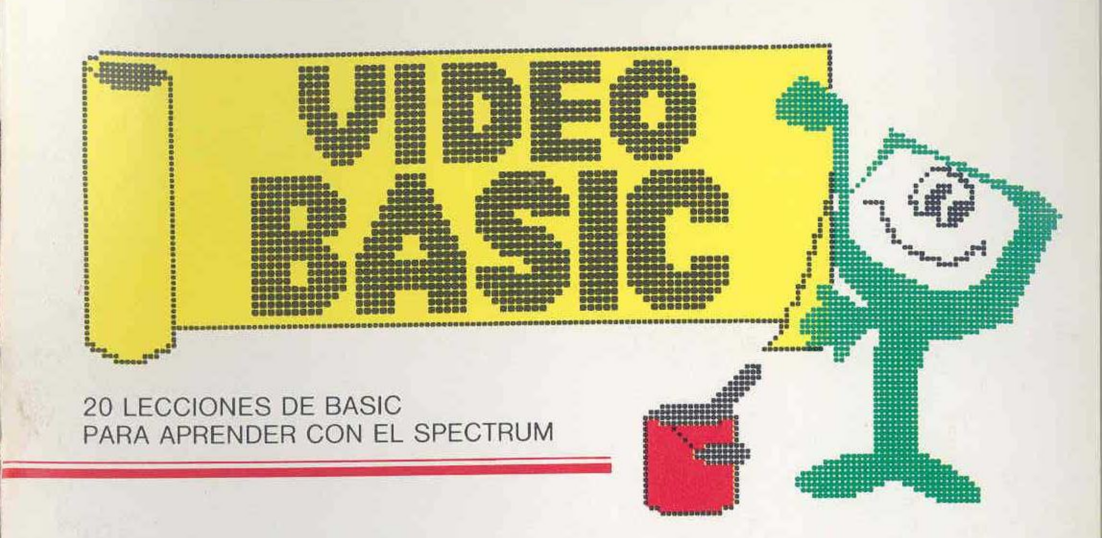


20 LECCIONES DE BASIC
PARA APRENDER CON EL SPECTRUM

INGELEK



JACKSON



4

Spectrum
16K/48K/PLUS

VIDEO BASIC
Una publicación de
INGELEK JACKSON
Director editor por INGELEK:
Antonio M. Ferrer
Director editor por JACKSON HISPANIA:
Lorenzo Bertagnolio
Director de producción:
Vicente Robles
Autor: Softidea
Redacción software italiano:
Francesco Franceschini,
Stefano Cremonesi
Redacción software castellano:
Fernando López, Antonio Carvajal,
Alberto Caffarato, Pilar Manzanera
Diseño gráfico:
Studio Nuovaidea
Ilustraciones:
Cinzia Ferrari, Silvano Scolari,
Equipo Galata
Ediciones INGELEK, S. A.
Dirección, redacción y administración,
números atrasados y suscripciones:
Avda. Alfonso XIII, 141
28016 Madrid. Tel. 2505820
Fotocomposición: Espacio y Punto, S. A.
Imprime: Gráficas Reunidas, S. A.,

ISBN del tomo 1: 84-85831-12-8
ISBN del fascículo: 84-85831-11-X
ISBN de la obra completa: 84-85831-10-1
Depósito Legal: M. 15.076-1985
Plan general de la obra:
20 fascículos y 20 casetes, de aparición quincenal, coleccionables .n 5 estuches.
Distribución en España:
COEDIS, S. A.
Valencia, 245. 08007 Barcelona.
INGELEK JACKSON garantiza la publicación de todos los fascículos y casetes que componen esta obra y el suministro de cualquier número atrasado o estuche mientras dure la publicación y hasta un año después de terminada.
El editor se reserva el derecho de modificar el precio de venta del fascículo, en el transcurso de la obra, si las circunstancias del mercado así lo exigen.
Junio, 1985.
Impreso en España.

INGELEK



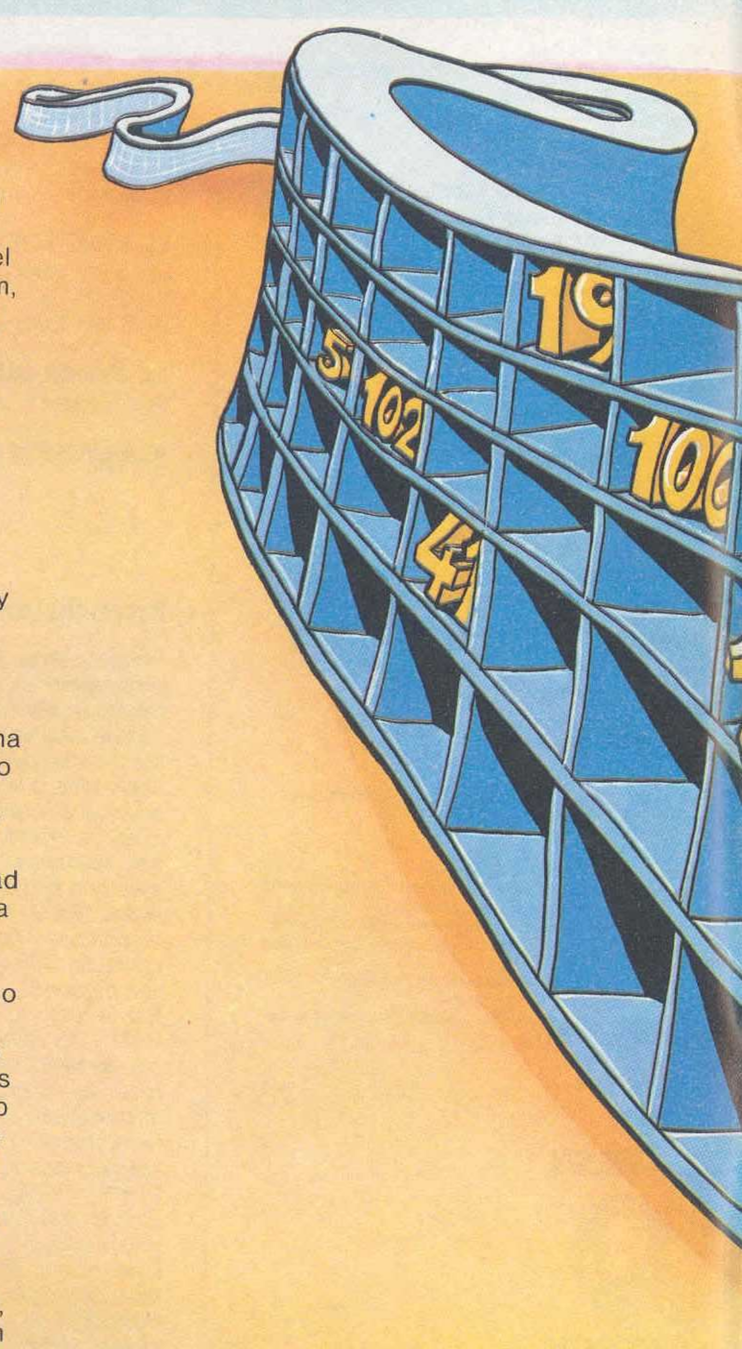
JACKSON

SUMARIO

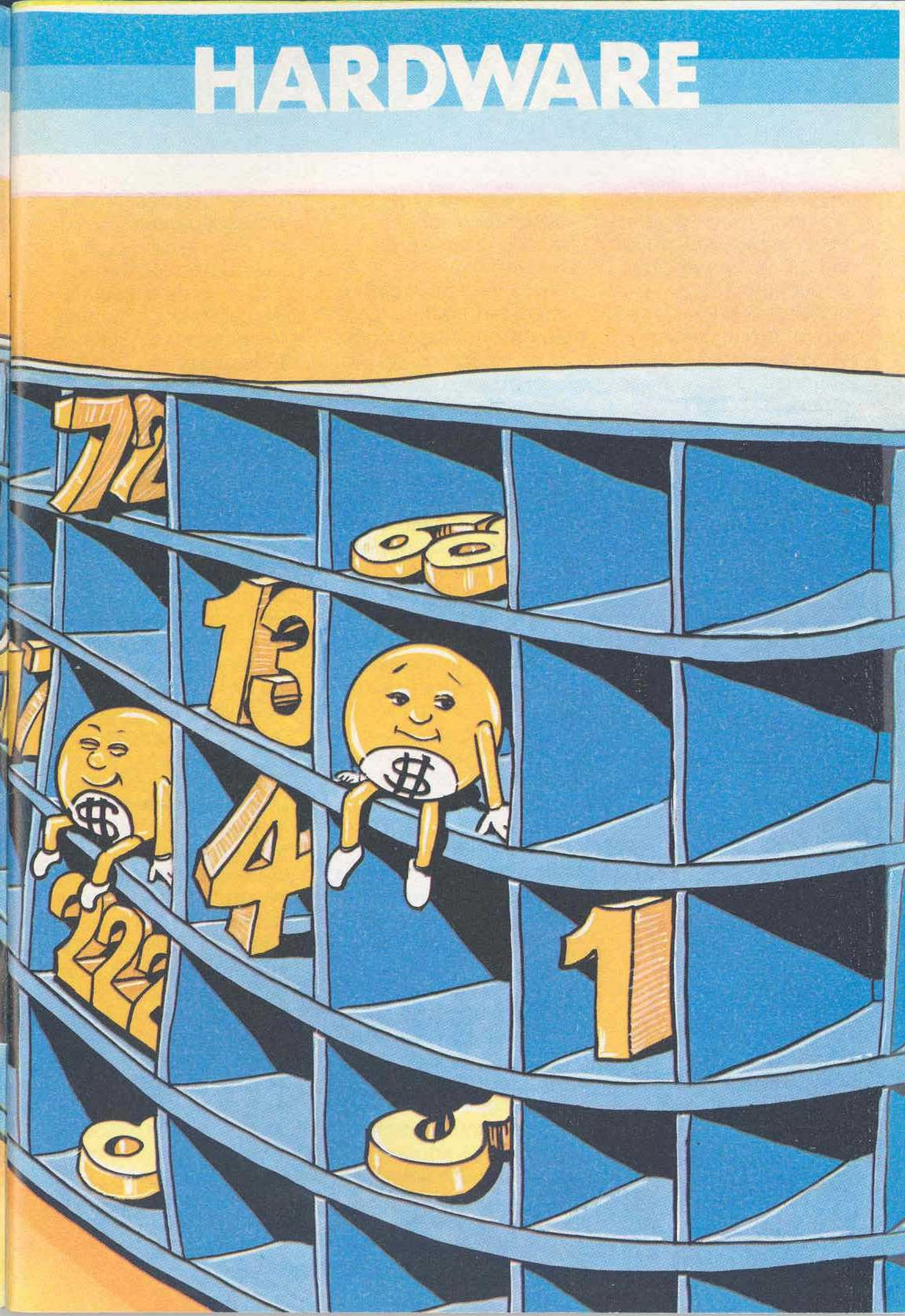
Introducción

Para nosotros (como para tu ordenador) es fundamental saber recordar: sin «memoria» tu ordenador es una caja vacía e inútil. ¿Quién le «recuerda» qué hacer cuando lo enciendes o le das las instrucciones BASIC? ¿Y dónde puede tomar apuntes (memorizar) sobre lo que esté haciendo? Sólo la memoria, una memoria prodigiosa que no olvide jamás (ROM) o que se encuentre disponible y receptiva (RAM) para cualquier información (siempre que sea numérica, se entiende).
Por lo tanto es necesario conocer las RAM y las ROM y el procedimiento que tu SPECTRUM emplea para recordar (escribir o leer) una información. ¿Pero cómo penetrar en la memoria? Tu ordenador tiene también una respuesta para esto: POKE y PEEK.
Existen además otras funciones, matemáticas y no matemáticas. En esta lección analizaremos algunas (INT, SQR, ABS) para terminar después con los contadores y los ciclos controlados.

La memoria



En las pasadas lecciones hemos visto cómo desarrolla el microprocesador, en el interior de tu Spectrum, la indispensable tarea de elaborar las instrucciones y las informaciones.
Lo que hoy nos proponemos entender es de que manera puede, una vez finalizadas las operaciones particulares, recordar y tener en cuenta las cosas ya hechas y aquellas que aún le quedan por hacer.
En realidad, el problema es bastante sencillo; no estando dotada de capacidades propias y autónomas de memorización, la unidad central (o CPU), emplea una memoria auxiliar, con la que se conecta cuando desea escribir o leer datos. De esta manera, a pesar de las modestas y elementales capacidades de cálculo de que dispone, puede ejecutar, gracias a una increíble velocidad, operaciones que en su conjunto resulten extremadamente laboriosas. Por lo tanto, es posible decir que en


HARDWARE

un ordenador la memoria representa una especie de almacén, en el cual se depositan o se recogen todas aquellas informaciones que el procesador considere como útiles o importantes a los fines de la ejecución de un programa.
(Ten en cuenta que todas las informaciones se expresan en forma de números, por lo tanto las memorias no son más que almacenes, más o menos grandes, de números).
Físicamente puedes imaginarte la memoria de un ordenador como si se tratara de una colmena microscópica: en cada celda (o posición) de este panal, la CPU puede escribir o leer un único dato a la vez.
Igualmente, cada posición de la memoria puede conservar, almacenada en su interior, una única información.
La unidad central localiza cada posición mediante una dirección numérica, a la que hace referencia cuando desea memorizar o encontrar cualquier dato.
El contenido de una posición de la memoria correspondiente a una cierta dirección nunca es una celdilla complementamente vacía.
Más o menos, es como si en cada posición estuviera contenida una lámpara (byte) con 8 bombillas (bit). Así, cuando la CPU desea modificar a un nuevo valor (por ejemplo 150 decimal, es decir 10010110 binario) el número representado por la lámpara situada en una determinada celda (por ejemplo la 1553), le es suficiente con apagar o encender las bombillas referentes a esa posición, hasta que la lámpara indique el nuevo valor (150).
Evidentemente, el número representado anteriormente por la lámpara de la celda 1553 se pierde, y es sustituido por el nuevo valor de 150 (10010110 en binario).
De igual manera, el procesador, cuando desea leer el contenido de una cierta dirección de la memoria, no hace más que copiar la combinación de bombillas encendidas correspondientes a la lámpara de la celda situada en esa dirección. Lo más importante que debes recordar sobre la memoria del ordenador es que ésta se localiza mediante direcciones y que las diversas posiciones contienen valores. Las direcciones indican una celda especifica (o posición) de la memoria. Cada posición contiene un solo valor a la vez.
Mientras que la dirección de una

posición siempre permanece igual, a veces el valor memorizado en una dirección puede, en cambio, ser modificado a placer.
Pero, como era natural esperarse, dado que no era materialmente posible que tu Spectrum pudiera contener en su interior millares de lámparas, los proyectistas de la casa constructora tuvieron que resolver el problema del mantenimiento de las informaciones recurriendo a otros medios.
Paremos un momento. Como habrás visto, el concepto de «recuerdo» para tu Spectrum es más bien sencillo; pero no se puede decir lo mismo para su respectivo problema técnico. La electrotecnia pura y la electrónica lo han resuelto, no sólo proponiendo cada vez soluciones más eficientes, sino también distintos dispositivos de memoria, tanto por sus propiedades como por sus posibilidades.

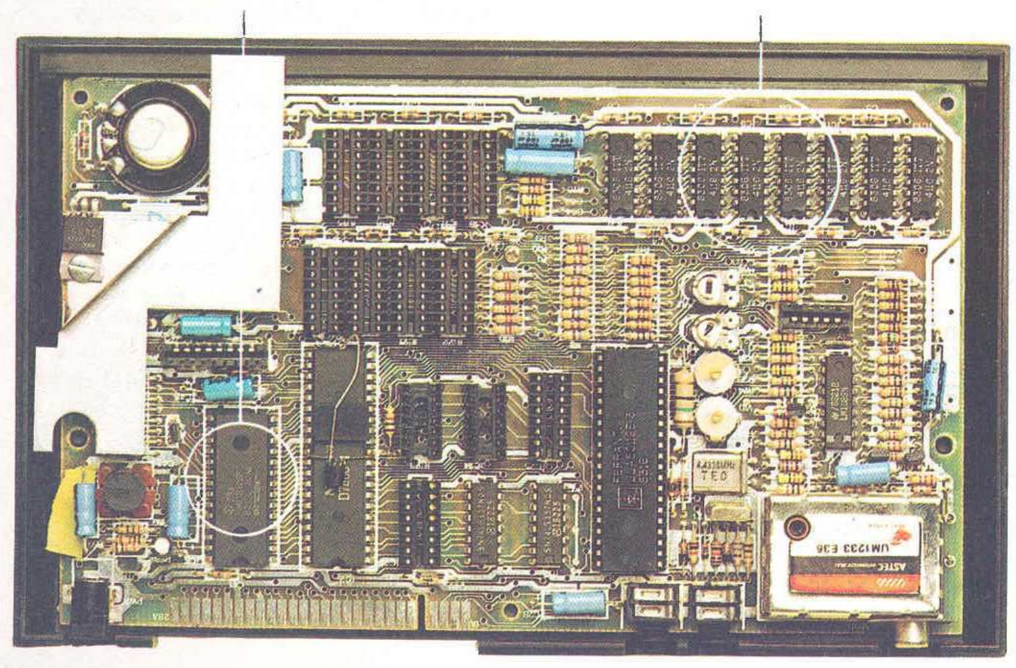
RAM y ROM

Aunque, como ya se ha dicho, toda la memorización se efectúe bajo la forma de bits (en palabras pobres, a traves de adecuadas secuencias de interruptores encendidos/apagados), existen substanciales diferencias entre unas memorias y otras.
Las dos grandes familias son:

- Las ROM (Read Only Memory-Memorias de solo lectura), que son permanentes: una vez impresionadas son inmutables, igual que una pelicula fotográfica;
- Las RAM (Random Access Memory-Memorias de acceso aleatorio), que son memorias no permanentes, llamadas también volátiles, en las cuales la información puede ser cambiada a placer.

La ROM es como el motor de un automovil, y la RAM como su depósito; es suficiente con que falte uno solo de los dos para que el vehiculo sea inutilizable.
Sin la ROM, tu Spectrum (y la CPU) no sería capaz de realizar ninguna tarea: no habría nadie que le recordara de forma permanente todo lo que debe de hacer. Sin la RAM no sabría donde colocar el resultado de sus elaboraciones, aún siendo capaz de tratar los datos.
En tu Spectrum las cèldas que constituyen la memoria son 65536 si posees un modelo de 48K o un Spectrum Plus, mientras que serán 32768 si en cambio tienes el de 16K.
A cada una de ellas le corresponde una dirección, es decir, un número coprendido entre 0 y 65536. Cada posición puede contener un byte de 8 bit: por lo tanto, el valor almacenado en cada celda es pues un número comprendido entre 0 y 255 (en efecto este último es el valor máximo representable por un número binario de 8 bit).
La memoria RAM sirve para contener los programas y los datos que se van escribiendo y usando por parte del procesador, cuando le hayas pedido que ejecute cualquier operación.

Esta es llamada de «acceso aleatorio», puesto que es posible acceder a una posición especifica, con la misma facilidad con la que se podria acceder a cualquier otra.
Una última característica, pero significativa de la memoria RAM es siguiente: cuando apagas el ordenador, todo lo que estaba contenido en ella será borrado y olvidado definitivamente.
Es por esta razón por lo que se recurre a dispositivos periféricos de memorización (que no forman parte del verdadero ordenador), como las grabadoras: éstas son capaces de conservar sobre cinta magnética las informaciones y programas que de otra manera se perderian.
En la memoria ROM, en cambio, los datos y los programas son grabados directa y permanentamente por la casa constructora. En ella están contenidas todas aquellas informaciones que el microprocesador necesita conocer en el momento del encendido del ordenador o en el momento de la ejecución de un programa. Por lo tanto, normalmente la ROM es utilizada para memorizar aquellos programas y aquellas informaciones que no deban sufrir ninguna variación en el tiempo.
Por ejemplo, el intérprete BASIC reside permanentemente en una memoria ROM, asi estará siempre listo, apenas se encienda tu Spectrum, para desempeñar



con diligencia su propio trabajo.
En tu Spectrum existen 16384 celdillas (correspondientes a 16K) celdillas empleadas como memoria ROM, es decir reservadas a aquellos datos y programas indispensables para el funcionamiento del sistema.
Por lo tanto, recuerda: la ROM y la RAM constituyen la memoria total sobre la que actúa el Spectrum; la primera contiene aquellos programas que ponen al ordenador en condiciones de fucionar en el momento del encendido y que por lo tanto no se pueden perder cuando apagues el ordenador. La RAM, en cambio, contiene los programas y los datos que sucesivamente escribes y empleas y que «abandonan» la memoria cuando dejan de ser útiles y necesarios; queda para tí el juzgar cuando merezca la pena evitar que estos últimos se pierdan irremediablemente, procediendo, por lo tanto, a guardarlos en cintas o en disco antes de desenchufar la alimentación de tu ordenador.

Como «recuerda» un ordenador

Nuestra memoria recuerda los acontecimientos que más nos han chocado, o sea, aquellos que nos han dejado una huella. En un computador la idea substancial es la misma: para memorizar alguna cosa es necesario producir una huella, un cambio. Imagina un tablero de ajedrez con una posición cualquiera, como si hubieras interrumpido la partida a medias. En cierto modo, el tablero es una memoria: mantiene la información (el orden de las piezas) hasta nueva orden.
Transformémoslo todo a términos electrónicos. La información (1 y 0) contenida en el circuito (el tablero) es mantenida en cada casilla hasta que no ocurra una variación (desplazamiento de una pieza), que modifique su estado (de 1 a 0 y viceversa). Este es el principio general sobre el que se basan todas las RAM. En las ROM la única diferencia es que la fase de cambio ha sido inhibida. Veamos como se alcanza este objetivo.
Las primeras memorias RAM se construyeron con núcleos de ferrita, pequeños anillos de material ferromagnético (precisamente ferrita), iguales al número de informaciones a memorizar. Un anillo de este tipo se puede magnetizar (para enterdernos: convertirse en un imán) de diferente manera, según el sentido de la corriente que lo atraviese. Sin entrar demasiado en la teoria del electromagnetismo, te será suficiente saber que:

- A cada estado de memorización se le asocia un valor binario (0 ó 1).
- Hay hilos verticales y horizontales (matriz) a través de los cuales es posible, enviando corriente, modificar el estado de un núcleo. De hecho, cada núcleo está colocado en la intersección de un hilo vertical con uno horizontal. Es un poco como jugar a la batalla naval: dando las coordenadas (un hilo horizontal y otro vertical) se localiza exactamente una

casilla (núcleo) del que se puede cambiar el estado (0 ó 1).
- Existe un hilo de lectura que, atravesando todos los anillos de la matriz, «lee» la información sin alterarla.

Se trata de circuitos conceptualmente sencillos, pero cuya realización comportaba no pocos problemas, tanto de naturaleza económica (alto coste) como de espacio. Para hacerte una idea, con una memoria de núcleos, es necesario un espacio igual al de un cubo de 20 cm. de lado para memorizar 4096 bytes. Por lo tanto, eran dos las exigencias: comprimir la mayor cantidad de informaciones en el menor espacio posible y lograr reducir los costes.

Nacieron así dispositivos con semiconductores: primero los transistores y después los circuitos integrados (chips).

Cada chip está constituido por una minúscula placa de silicio sobre la que han sido colocados —con especiales y sofisticados procedimientos— todos aquellos dispositivos electrónicos (diodos, transistores, resistencias, etc.) necesarios para el funcionamiento del circuito. Las piezas de plástico negro con muchas patillas (pines) de metal, que podrás ver levantando la tapa de tu Spectrum, no son más que los recipientes de tales chips.

Entre las memorias a semiconductores es necesario distinguir:
- Memorias estáticas; cada bit es memorizado por un circuito biestable (que puede asumir dos estados estables) y permanece sin problemas hasta una eventual caída (falta) de tensión de alimentación.
- Memorias dinámicas; los bits se memorizan en condensadores microscópicos, que se descargan lentamente y requieren un continuo refresco (refresh) de la información.

Las informaciones contenidas en la memoria son repetidas periódicamente (es decir, la memoria es impresa varias veces) para no perderlas.

En los primeros ordenadores, este trabajo lo desempeñaba la CPU que a intervalos de tiempo determinados (inferiores a la milésima de segundo), volvia a copiar todo el contenido de la memoria.

A primera vista pudiera parecer que la CPU tiene que desarrollar demasiados cometidos; sin embargo, el refresco de memoria no perjudica a la actividad de la CPU más de lo que el hipo pudiera fastidiar a un ser humano.

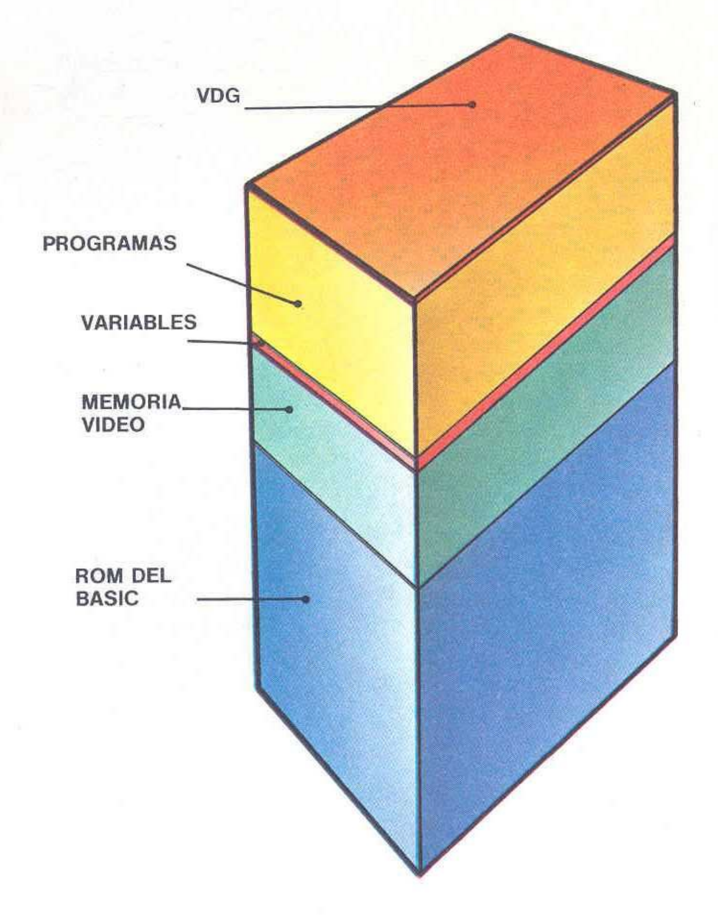
Periódicamente se interrumpen las actividades normales, pero entre un golpe de

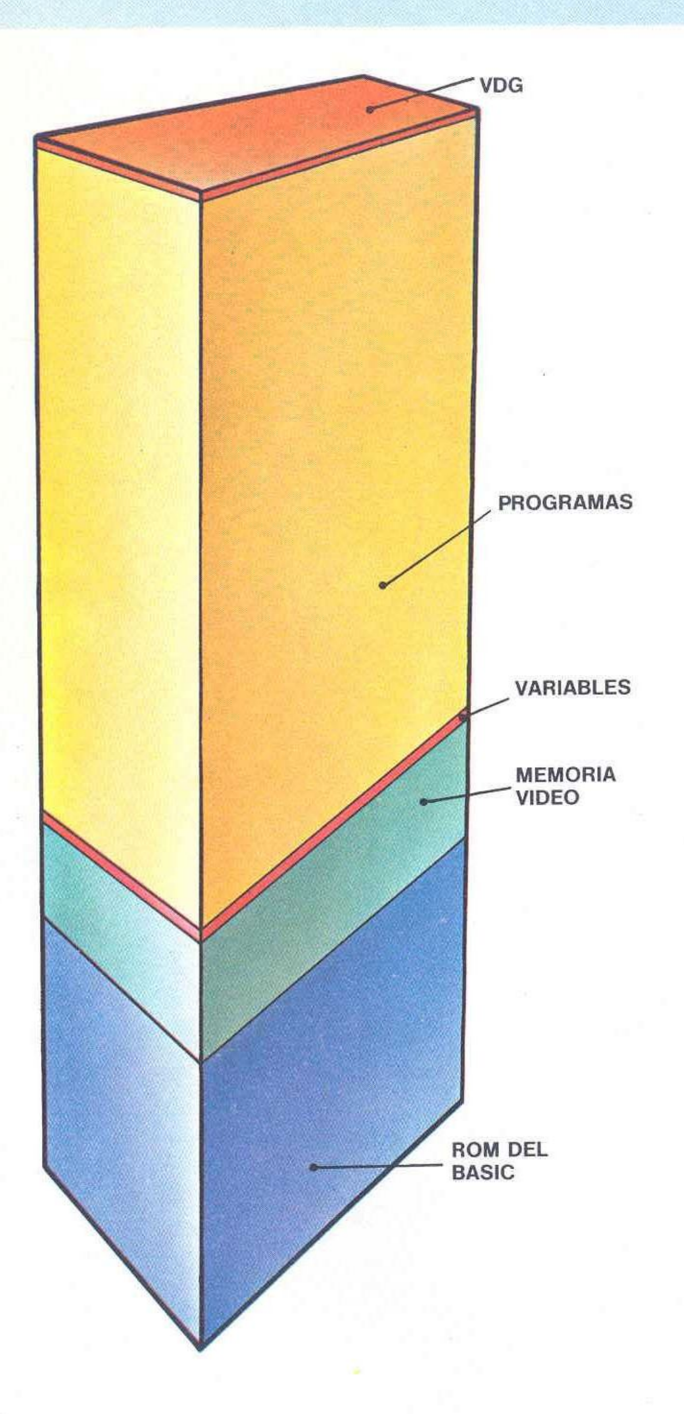
hipo y el siguiente queda libertad para hacer lo que se desee. Y el refresco, para una CPU que tenga frecuencia de reloj en el orden del MHz, no es más que un golpe de hipo. Un MHz (Megahertzio), significa que el reloj del ordenador realiza un millón de operaciones al segundo. ¿Rápido, verdad?
Este último tipo de memoria RAM es el que se emplea en tu Spectrum, y en general en todos los demás ordenadores personales.
Veamos ahora las ROM. Son memorias no volátiles, como ya se ha dicho mantienen permanentemente las informaciones. Están formadas básicamente por una red de semiconductores, es decir, una serie de conductores organizados en filas y columnas. Cada intersección entre filas y columnas representa un bit a nivel lógico 0 ó 1, según que línea y columna se unan o no entre ellas. La programación (es decir, la realización de la red) se realiza en base a las exigencias de quien la ejecuta, y una vez establecida es definitiva (no puede ser modificada de ninguna manera).
Existen también otros tipos de memoria de sólo lectura: las más empleadas son las PROM y las EPROM.
Las PROM (Programable Read Only Memory) son, como su nombre indica, memorias de sólo lectura, programable por el usuario gracias a dispositivos adecuados (los programadores de PROM). Se usan para objetivos especiales, fundamentalmente cuando se necesiten ROM para piezas únicas, para prototipos, pruebas, máquinas especiales y en cualquier circunstancia en general donde el uso de una ROM fuera demasiado costoso porque se empleará únicamente para un número pequeño de piezas. De hecho el uso de ROM es conveniente cuando la cantidad de piezas supere el millar.
Las EPROM (Erasable Programmable Read Only Memory), también son de sólo lectura, con la ventaja de ser programables más de una vez mediante el uso de aparatos especiales.

Los mapas de memoria

Hasta ahora hemos explicado de que manera y mediante qué sistemas se memoriza la información. Pero ¿de qué modo se lee? Y, sobre todo ¿cómo saber que información debo tomar y dónde?
Imagina que te encuentres en una gran biblioteca, dotada con decenas de miles de volúmenes. ¿Cómo organizar el archivo para poder encontrar rápidamente un libro?
Una solución aceptable es numerar progresivamente los volúmenes, para acelerar al máximo la búsqueda de un libro.
En un ordenador la solución es similar: cada celda de memoria (es decir, aquella zona que contiene una «palabra» o byte) está identificada por un número (la dirección).
Normalmente, la cantidad de memoria que un ordenador debe mantener bajo control es de aproximadamente 64K bytes, es decir, 65536 bytes (en decimal, el símbolo K equivale a mil, pero en binario es 1024, equivalente a 2^{10}).

Direccionar significa otorgar un número (una dirección), a cada celdilla de memoria (equivale a la numeración de los volúmenes), definido en el interior del ordenador en forma única y constante. Además, vista la heterogeneidad de los datos que el ordenador debe tratar (los libros para nuestro bibliotecario), quien proyecta la máquina los divide en secciones (materias).
Se crea así el llamado mapa de memoria, es decir, la organización de los datos en ella contenidos, prevista para agrupar a todos aquellos bytes que tengan algo en común.
El mapa de memoria no es más que una explicación detallada sobre la presencia y disposición de las direcciones previstas para las diversas tareas, que la casa constructora proporciona al ordenador.
Siendo estas tareas múltiples y a veces complejas, el mapa de memoria de tu Spectrum podrá parecerte algo difícil. Pero no te preocupes: ¡el mapa existe para permitirte ignorarlo! Precisamente para hacer posible el desarrollo de las distintas tareas por





parte del ordenador sin que tu debas ocuparte de ellas.
Los elementos más importantes del mapa de memoria del Spectrum son:

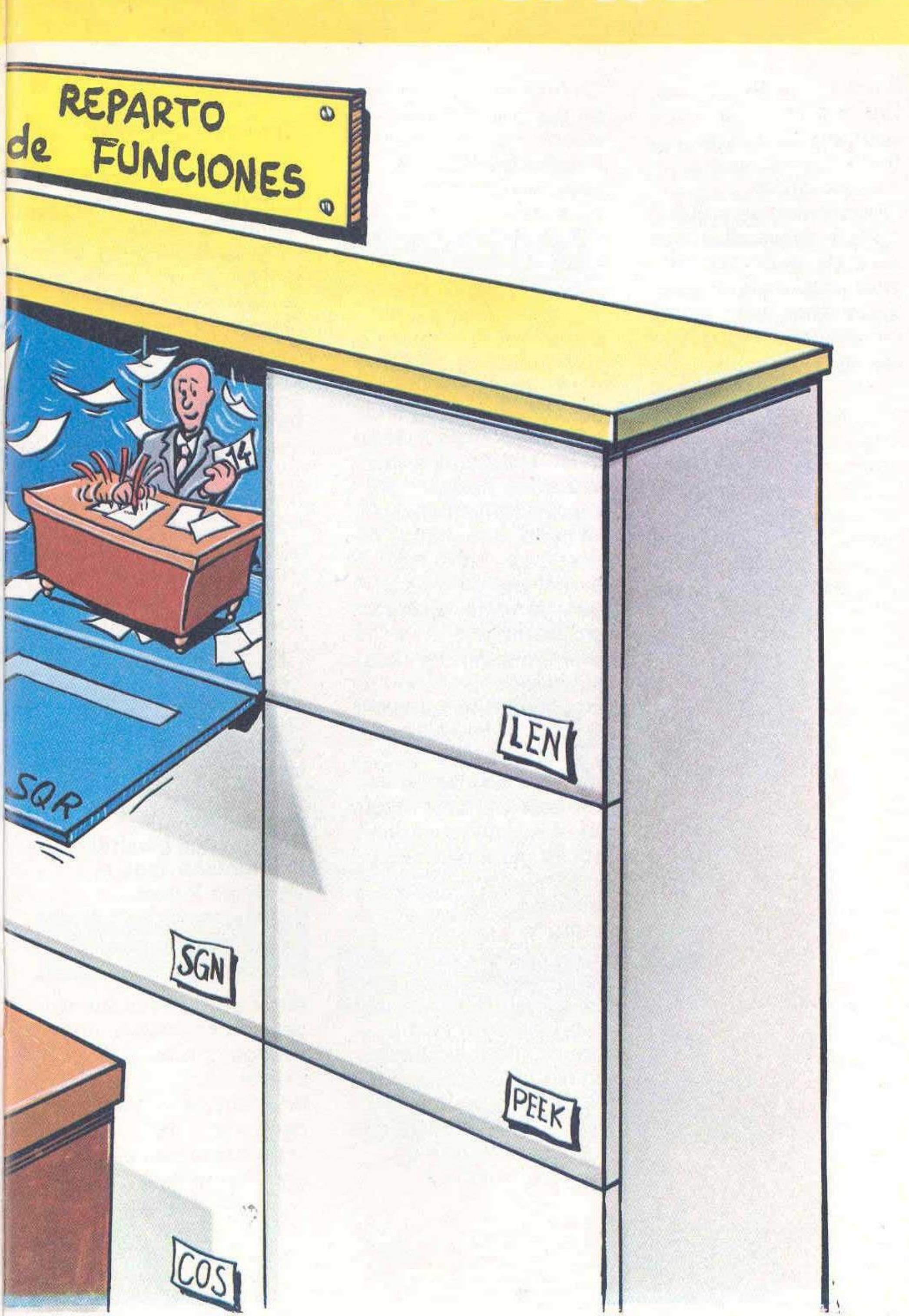
- La ROM del intérprete BASIC y del sistema operativo (16K), situada entre las direcciones 0 y 16384. Aquí reside prácticamente todo el «saber» de tu Spectrum.
- La memoria de video o display file (entre 16384 y 22528) y la memoria de atributos (entre 22528 y 23296) donde están memorizados los datos referentes a las imágenes visualizadas.
- El área de los programas BASIC (desde la dirección 23765 en adelante) en la que se memorizan los programas en BASIC.

Las figuras representan los mapas de memoria del Spectrum, es decir, dónde están o dónde se memorizan y organizan las informaciones.Obviamente el Spectrum 48K, tiene un área de programas mucho más extendida que la del 16K.

Las funciones

Coge un momento con la mano una calculadora de bolsillo. Muy probablemente sea capaz de resolver, además de las operaciones aritméticas corrientes, también cálculos más complicados y dificultosos que sencillas restas o multiplicaciones: por ejemplo, la extracción de raices cuadradas o la elevación de un número a potencias. En BASIC existen algunas funciones que te permiten obtener con tu Spectrum los mismos resultados que con una calculadora: las




REPARTO de FUNCIONES
14
LEN
SQR
SGN
PEEK
COS

funciones matemáticas. Una función es un operador al que le proporcionas una o más variables independientes y que ejecuta sobre ellas una serie de elaboraciones para proporcionar el único valor que constituye el resultado (variable dependiente).

En terminos generales se escribe: y = f (x), donde x es la variable independiente e y la dependiente. Por ejemplo:

```
LET A = SQR (8)
```

significa: asigna a la variable A el valor de la raíz cuadrada de 8. SQR() es la función que ha sido empleada en este caso: proporciona como resultado la raíz cuadrada de la expresión (también llamada argumento) encerrada entre los paréntesis.
Naturalmente, distintos argumentos proporcionan distintos resultados. Si el argumento no es válido, la función no se ejecuta y se conecta la visualización de un mensaje de error. Así, si en el ejemplo anterior le hubiéramos pedido al ordenador que ejecutara esta instrucción:

```
LET A = SQR (–8)
```

habríamos obtenido como único resultado un INVALID ARGUMENT, es decir, argumento no válido: ¡No es posible la raíz cuadrada de un número negativo!

La sintaxis de las funciones

Las funciones, como ya te habrás imaginado, forman parte del lenguaje del Spectrum: residen en una zona de la ROM a la que se refiere el ordenador todas las veces que pretendas usar alguna. Por lo tanto, para referirte a las distintas funciones debes emplear la palabra reservada que la casa constructora haya asignado a cada una de ellas: normalmente es una abreviatura del correspondiente término inglés (por ejemplo: SQR es la abreviatura de SQuare Root, raíz cuadrada).
Cualquier error de tecleado o de sintáxis será localizado inmediatamente.
En terminos generales una función tiene el siguiente formato:

```
función (argumento)
```

Cada argumento puede ser una constante, una variable o una expresión.
Una función presente dentro de una instrucción es calculada anteriormente a

cualquier otro operador. El argumento de una función, puede ser otra función.
Las funciones nunca pueden aparecer a la izquierda del signo de igual. Una instrucción de este tipo:

```
SQR (B) = A
```

constituye el vehículo más seguro para provocar el envío de un mensaje de error por parte de tu Spectrum. En su conjunto, las funciones disponibles en tu Spectrum, y no únicamente matemáticas, son aproximadamente una veintena: paso a paso aprenderás a conocerlas todas y a emplearlas con facilidad.

SQR

Ya has conocido la primera de estas funciones: es SQR (abreviación, como ya se ha dicho, de SQuare Root, raíz cuadrada), la amiga que te devuelve la raíz cuadrada del número que tú le hayas proporcionado como argumento.
Lo único que ella exige es que el argumento sobre el que tenga que operar sea positivo, o como máximo, nulo. En el caso contrario el intérprete BASIC te envía una señal de error (INVALID ARGUMENT), parando la ejecución.
Supongamos que deseemos calcular y visualizar en pantalla la raíz cuadrada de 3; la instruccción; por lo tanto, deberá ser:

```
PRINT SQR (3)
```

el argumento entre paréntesis puede ser cualquier expresión «legal» del BASIC (es decir, aceptable por el intérprete). Un comando de este tipo:

```
PRINT SQR (3 * 4  – (SQR (16/4))),
```

aunque no te parezca de comprensión inmediata, es perfectamente admisible en el interior de un programa o de una instrucción.
Tu Spectrum es exacto, pero no exacto en sentido absoluto; de un número, imprime como máximo 8 ó 9 cifras, mientras que conserva, y considera, en memoria 10. En efecto, cualquier método que uses para el cálculo de la raíz cuadrada, o de otras operaciones complejas, dará siempre un resultado que será una aproximación al valor real. Tenlo en cuenta.

Ejemplos

LET A = SQR (5.5 + 3.5)	Asigna a la variable A el valor de la raiz cuadrada de 5.5 + 3.5, es decir, 3.
LET B = SQR (20 – SQR (16))	La raiz cuadrada de 16 es 4: 20 números menos 4 da 16. A la variable B se le asigna por lo tanto el valor 4.
LET C = SQR (NUMERO)	NUMERO es una variable, iPero cuidado, en caso contrario ésta instrucción provocará un error!
LET D = 4 * SQR (9) – 2	Asigna a D el valor 10, obtenido resolviendo la sencilla expresión a la derecha del =. Como habrás podido observar, si en la misma expresión aparecen juntas funciones y operaciones aritméticas, las funciones son las primeras que se calculan. Por lo tanto, su prioridad de ejecución es la máxima. Pero como de costumbre, el orden de los paréntesis puede modificar el orden de ejecución de los cálculos.
LET NUM1 = SQR (25) – SQR (256)	Esta instrucción asigna a NUM1 el valor 5 – 16, es decir, – 11.
LET NUM2 = SQR (25 – SQR (256))	Los paréntesis han modificado el orden de prioridad: por lo tanto la instrucción asigna a la variable NUM2 el valor de la raiz de 25 – 16, es decir, 3.

INT

INT es otra función disponible en tu ordenador. Significa: parte entera del argumento. Proporciona como resultado el número entero (aproximado por defecto) más cercano al argumento real que hayas encerrado entre paréntesis.
Por lo tanto:

```
INT (34.125) = 34
```

El uso de la función INT aplicada a los números negativos con decimales, proporciona el número negativo entero inmediatamente inferior al argumento:

```
INT (−4.41) = −5
```

Ten pues en cuenta que mientras que en el caso de los números positivos el resultado de la función INT es equivalente al valor del argumento, privado de su parte decimal, cuando el argumento es un número negativo tienes que pensar en el valor inmediatamente menor de la expresión de salida.
Son muchas las posibles formas de uso de la función INT: por ejemplo, los bancos, cuando a fin de año calculan los intereses producidos por las libretas de ahorro, redondean todos los importes precisamente mediante esta función.
Además, con el uso de INT puedes separar la parte entera y la parte decimal de un número real.
Estas instrucciones:

```
LET B = INT (A)
LET C = A − INT (A)
```

consiguen que se asigne a las variables B y C, las partes entera y decimal del argumento constituido por la variable A.





En el ejemplo siguiente podrás observar como la función INT parece estar hecha para los bancos y no para sus clientes... bromas a parte, espero que te pueda esclarecer el efecto de INT sobre los números decimales.

```
 10 REM PRIMERA VENTAJA (SOBRE LOS DEPOSITOS)
 20 LET AF = 985000.9
 30 REM AF = AHORRO AL FINAL DEL AÑO
 40 LET AN = INT (RF): PRINT AN
 50 REM TU AHORRO HA BAJADO EN...
 60 SEGUNDA VENTAJA (SOBRE LOS PRESTAMOS)
 70 LET DF = – 1450.1
 80 REM DF = DEUDA A FINAL DE AÑO
 90 LET DN = INT (DF): PRINT DN
100 REM TU DEUDA HA AUMENTADO EN...
```

Ejemplos

```
LET X = 33
LET Y = 100
LET Z = INT (Y/X)
```

Asigna a la variable numérica Z la parte entera del cociente entre los valores contenidos en las variables numéricas X e Y (Z, por lo tanto, vale 3).

```
LET NUMERO = SQR (INT (100.12))
```

La variable NUMERO toma el valor de la raiz de 100, es decir 10.

```
LET A = 21.5
LET B = 22
LET C = INT (A – B)
```

Dado que el valor del argumento de la función es un número decimal negativo (en efecto, A – B da – 0.5), la variable C toma el valor entero inmediatamente inferior a él (o sea, –1).

SGN

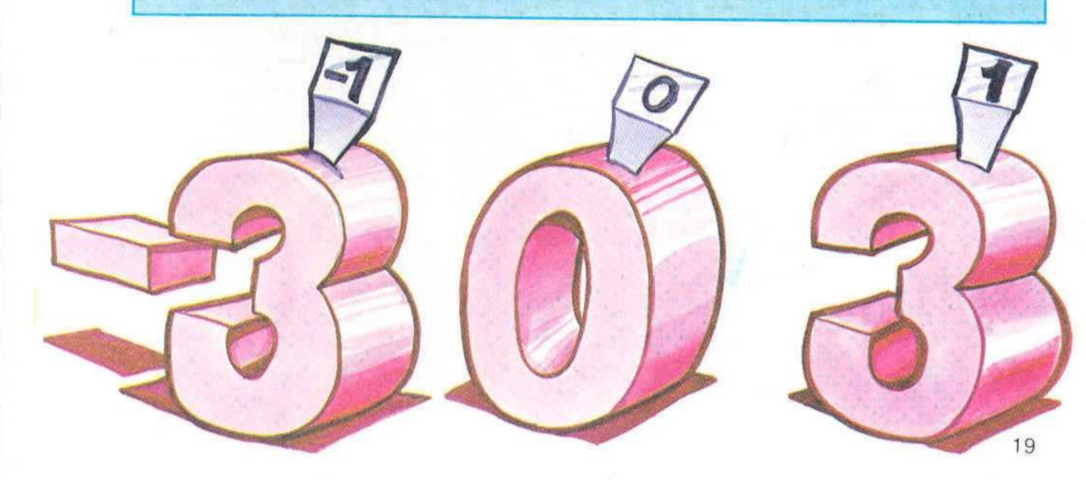
Muchas veces resulta útil conocer si una variable (o un número, o bien una expresión) es positiva, negativa o nula: la función SGN (del inglés sign), también llamada signo, proporciona la solución a este problema.

Sea el que sea el número inicial (entero o decimal), ésta proporcionará como resultado:
1, si el valor del argumento es positivo;
0, si el valor del argumento es nulo;
−1, si el valor del argumento es negativo.
Por ejemplo:

```
LET AA = SGN (33.25)
LET BB = SGN (4 − SQR (16))
LET CC = SGN (−13.2)
```

Estas instrucciones asignan respectivamente a las variables AA, BB, CC los valores 1, 0 y −1.
La función signo desempeña un papel fundamental en la programación. No tiene tiene un auténtico efecto, pero «señala» en el interior de un programa una determinada característica (el signo precisamente).
Su función principal es precisamente la de «advertir» al propio programa del cambio de signo de una expresión. El siguiente programa es un claro ejemplo de lo dicho:

```
10 REM VERIFICACION FINANCIERA CON SGN
20 INPUT «¿CUANTO DINERO TENIAS AYER?»; DA
30 INPUT «¿CUANTO DINERO TIENES HOY?»; DH
40 LET DIF = DA − DH
50 IF SGN (DIF) = 0 THEN PRINT «NINGUNA VARIACION»
60 IF SGN (DIF) = −1 THEN PRINT «TIENES MAS DINERO»
70 IF SGN (DIF) = 1 THEN PRINT «HAS GASTADO»
```



ABS

ABS es otra función numérica. Su tarea es la de convertir el valor del argumento en un número positivo, independientemente del valor que éste poseyera. ABS es la abreviatura del inglés ABSolute Value (valor absoluto).
Por ejemplo:

```
ABS (–3.41) = ABS (3.41) = 3.41
```

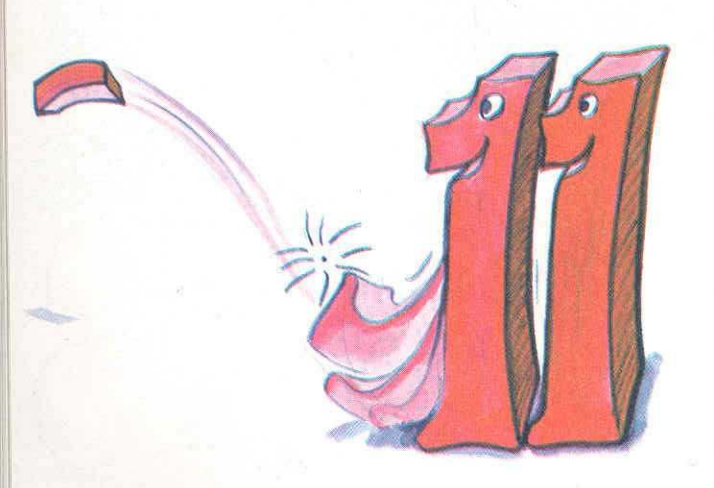
El empleo de esta función resulta especialmente útil cuando se desean efectuar cálculos sin tenerse que preocupar de los diferentes signos. Así, para estar seguros de que una expresión de este tipo:

```
LET C = SQR (TOTAL)
```

sea correcta en cualquier caso, es suficiente con escribirla así:

```
LET C = SQR (ABS (TOTAL))
```

Actuando de este modo, el signo de la variable TOTAL ya no influirá de una forma tan drástica el éxito de la operación, salvaguardándonos de posibles errores y de paradas en el programa.
Se intuye rápidamente como opera tu Spectrum para obtener la función ABS: basta con poner el signo + en la posición de memoria que contiene el signo del número deseado. El valor absoluto considera los números únicamente como positivos.
Un excelente ejemplo del uso que esta función puede



desempeñar en el interior de un programa, es el cálculo de la longitud de un segmento en el plano.

Imaginemos un plano cartesiano donde X1, Y1 y X2, Y2 son los extremos del segmento del que debemos determinar la longitud, la fórmula a emplear es:

$$\sqrt{|X1 - X2|^2 + |Y1 - Y2|^2} = \text{LONGITUD DEL SEGMENTO}$$

donde los || representan matemáticamente el valor absoluto de la expresión contenida en su interior.
He aquí el programa:

```
10 REM CALCULA LA LONGITUD DE UN
   SEGMENTO EN EL PLANO
20 INPUT «ESCRIBE LOS EXTREMOS DEL
   SEGMENTO A LO LARGO DEL EJE X"; X1, X2
30 INPUT "ESCRIBE LOS EXTREMOS DEL
   SEGMENTO A LO LARGO DEL EJE Y"; Y1, Y2
40 REM CALCULO DE LA LONGITUD
50 LET LONGITUD = SQR (ABS (X1 — X2) ↑ 2 +
   (ABS(Y1 — Y2) ↑ 2)
60 PRINT "LA LONGITUD DEL SEGMENTO ES";
   LONGITUD
70 INPUT "¿QUIERES QUE CALCULE OTRO?
   (S/N)"; A$
80 IF A$ = "S" THEN GOTO 20
90 END
```

PI (π)

PI no es una auténtica función: más bien se trata de una constante que ha sido memorizada permanentemente con este nombre en el interior de tu Spectrum. Su objeto es muy simple: evitar que todas las veces que este número sea necesario, te veas obligado a teclear una a una las varias cifras que lo componen. Para ver su efecto será suficiente con que teclees:

```
PRINT PI
```

Verás aparecer en pantalla el número 3.1415927

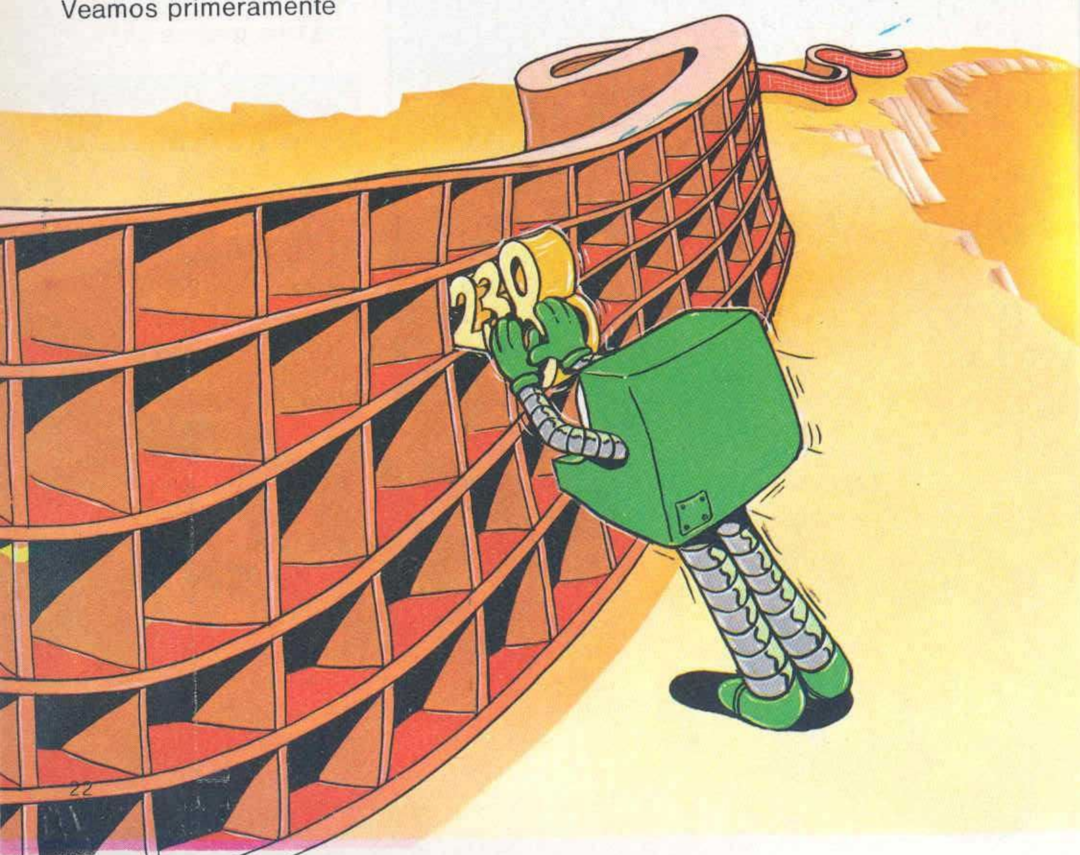


POKE

Dentro de tu Spectrum, en cada una de las miles de posiciones que componen su memoria, cada información se memoriza bajo la forma de un byte, es decir, mediante números comprendidos entre 0 y 255.
El intérprete BASIC te permite curiosear o modificar los valores presentes en cada una de las celdillas a través de PEEK y POKE.
Veamos primeramente POKE (que podría traducirse al castellano como: «meter las narices en»).
POKE sirve para memorizar directamente un número en una posición de la memoria, saltándose los mecanismos normales del BASIC.
Esto te permite escribir en una determinada posición de la memoria un número cualquiera, comprendido entre 0 y 255.
Por ejemplo:

```
POKE 3200, 100
```

seguido de ENTER, tendrá como efecto insertar en la posición de memoria con dirección número 32000 el valor del argumento, es decir 100. POKE no es una función porque no se requiere el uso de los paréntesis. El primer valor indica la celda de memoria, el segundo el valor a memorizar.
El valor que desees escribir en la memoria



debe ser menor o igual que 255: en el caso contrario, el Spectrum se parará, enviándote un mensaje para advertirte del error.

```
INTEGER OUT OF
RANGE
```

Naturalmente la orden POKE produce efecto únicamente sobre las posiciones de memoria pertenecientes a la memoria RAM: si la dirección se refiere a una posición de la memoria ROM, no ocurrirá nada a continuación de POKE. Recuerda: ¡La memoria ROM no es modificable!
Por lo tanto no temas, con una orden POKE (como también con cualquier otra orden BASIC) es absolutamente imposible estropear de forma definitiva e irremediable la memoria de tu Spectrum.
Tanto la dirección cómo el dato pueden proporcionarse bajo forma de expresión numérica o de variable. Por ejemplo:

```
POKE 2000 * 16, SQR (10000)
```

es completamente legal, así como también

```
LET I = 32000 : LET D = 100 : POKE I, D
```

Conviene decir que POKE acepta también números negativos, puesto que los convierte en positivos, sumándoles siempre 256.
En general todos los datos que el ordenador tiene en la RAM pueden ser modificados según las exigencias de quien programa. Es importante saber cuáles son los que hay que modificar para obtener el resultado deseado.

Estas informaciones están contenidas en las variables del sistema, números empleados por el ordenador para «acordarse» de determinadas operaciones. Por ejemplo, en tu Spectrum hay una celda de la memoria RAM en la que el sistema operativo, en el momento del encendido, escribe un número que le recuerda al ordenador cuanto tiempo, en cincuentaavos de segundo, es necesario tener pulsada una tecla antes de que se conecte la repetición automática.
La dirección de esa celda es la 23561, y el valor escrito por el sistema operativo, es decir, la variable del sistema, es 35.
Por lo tanto deben pasar 35 cincuentaavos de segundo antes de que empiece el «eco» de la tecla.
Gracias a POKE, tú puedes ahora modificar ese valor a tu gusto, inténtalo.
Introduce por ejemplo, en modo directo:

```
POKE 23561, 1
```

¿Has visto que diferencia?

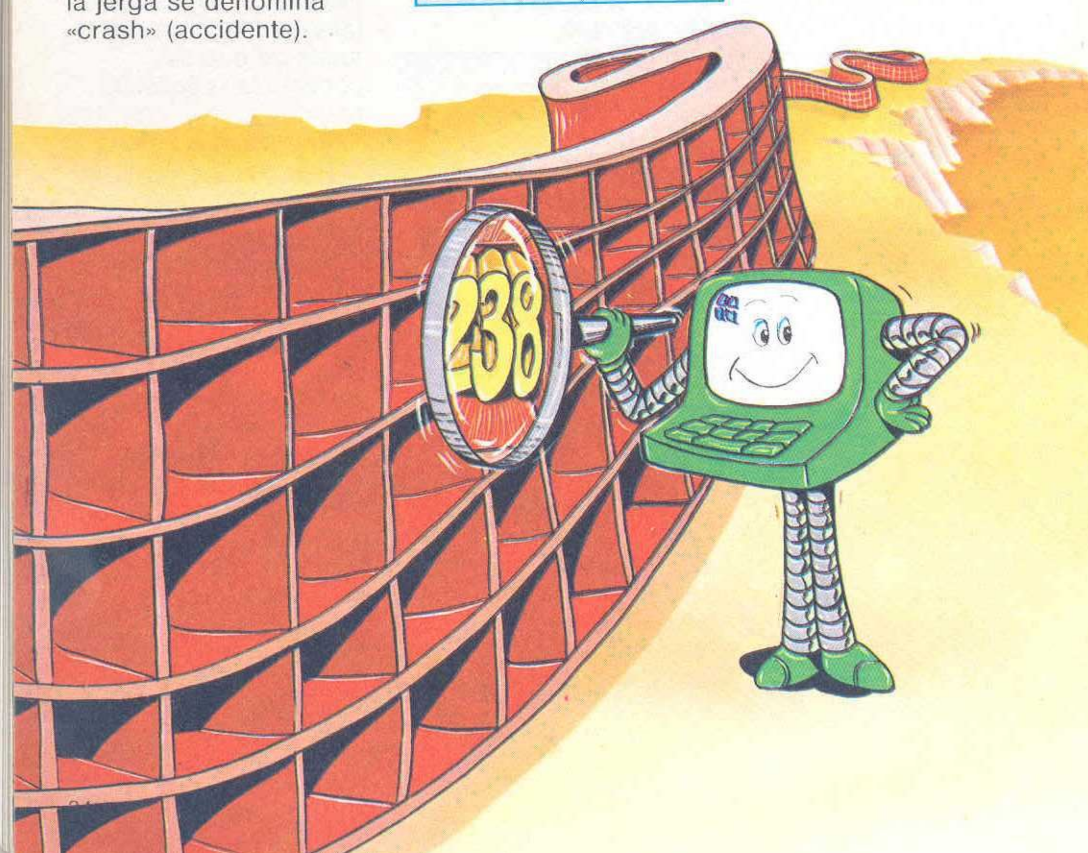
Prueba con otros valores.
Así pues, puedes modificar las variables del sistema. Pero ¡cuidado!, con no cambiar algunas variables clave, fundamentales para el ordenador, puesto que así causarías un bloqueo total del sistema, que por una orden impropia (la variable del sistema alterada), ya no sabe como seguir trabajando y se bloquea, lo que en la jerga se denomina «crash» (accidente).

PEEK

PEEK, (ojear, cotillear), es, en cierto sentido, el opuesto de POKE: esta función te permite conocer el valor contenido en una posición cualquiera de la memoria. Así, si has ejecutado anteriormente la sentencia POKE 32000, 100,
el resultado de la instrucción

```
PRINT PEEK (32000)
```

será el de imprimir en pantalla el número 100; PEEK no hará más que leer el contenido de la posición 32000, mientras que PRINT visualizará su valor en pantalla.
La función PEEK puede aplicarse tanto a la memoria RAM como a la memoria ROM, lo que es fácilmente comprensible, dado que la operación de lectura se puede ejecutar sobre cualquier posición de la memoria.



El argumento de PEEK, es decir, la dirección de la posición de la que se tenga que leer el valor, debe estar siempre comprendido entre 0 y el límite máximo de la memoria, 65535; una dirección fuera de estos límites provoca la parada del programa y la visualización de un mensaje de error.

INTEGER OUT OF RANGE

de PEEK y POKE podría parecer superfluo: ¿cuándo puede interesarte escribir o leer en una posición de memoria?
Sin embargo, como ya has leído, y te lo repito, se trata efectivamente de comandos extremadamente potentes y útiles. La memoria, en efecto, posee algunas direcciones mágicas; refiriéndote a ellas a través de PEEK o POKE, puedes obtener, por ejemplo, efectos gráficos y sonoros absolutamente impensables con otras órdenes BASIC.
Pero la otra cara de la moneda es que escribiendo por error un número equivocado en una dirección incorrecta, tienes la posibilidad de perder en un instante programas e informaciones que quizá te hayan supuesto horas de trabajo en el teclado.
Moraleja: si no estás más que seguro de lo que estás haciendo, intenta evitar —especialmente en el interior de programas largos— el uso de PEEK y especialmente de POKE: quizá tengas que renunciar a algún posible efecto coreográfico, pero tendrás la certidumbre de no haber introducido alguna sentencia que puede traer malas sorpresas y resultados indeseables.
Concéntrate y reflexiona sobre las analogías y diferencias entre:

```
—LET A = 37 : PRINT A
—POKE 30000, 37 : PRINT PEEK (30000)
```

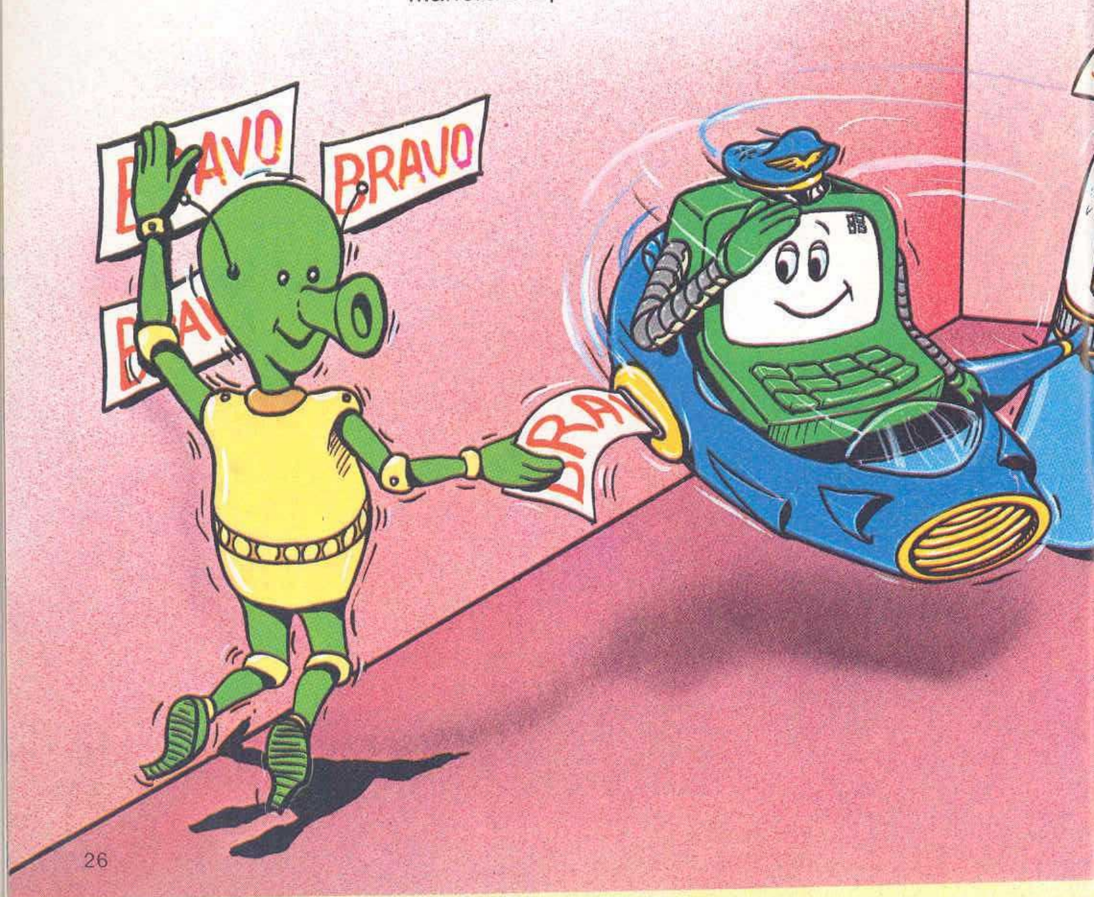
El contador y los ciclos controlados

Existen muchos casos y situaciones en los cuales un ordenador es llamado a ejecutar, durante un cierto número de veces, la misma operación.

Supongamos por ejemplo que deseemos imprimir todos los números enteros comprendidos entre 1 y 50. Un posible programa sería:

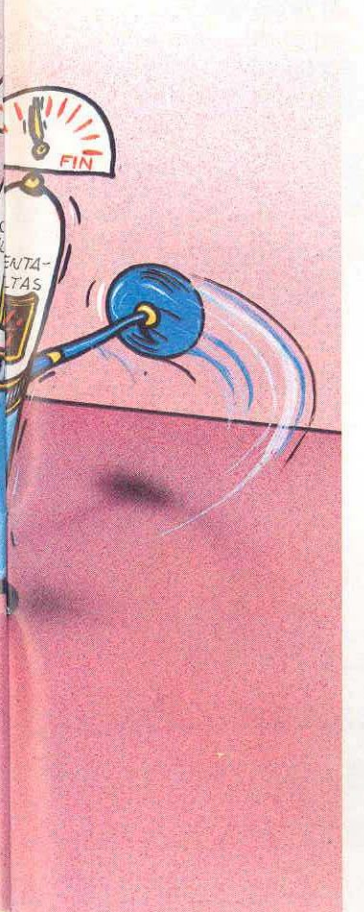
```
10 PRINT 1
20 PRINT 2
30 PRINT 3
40 PRINT 4
```

y así sucesivamente. Te darás rápidamente cuenta de que esta manera de proceder es ineficaz e inadmisible: el trabajo de tener que escribir 50 instrucciones para resolver una tarea tan banal, resultaría seguramente excesivo. Pero, afortunadamente, existe una manera más sencilla y racional para afrontar el problema: emplear un contador. El contador no es más que una variable empleada para contar una determinada secuencia de



instrucciones/acciones ejecutadas por el ordenador. Mediante esta variable se hace así posible controlar el número de veces durante las cuales tales acciones/instrucciones son llevadas a cabo. En nuestro caso el problema podría ser escrito de este modo:

```
10 LET NUM = 1
```



Esta línea define la variable NUM (el contador) y establece su valor en 1.

```
20 PRINT NUM
```

La línea 20 imprime el contador.

```
30 LET NUM = NUM + 1
```

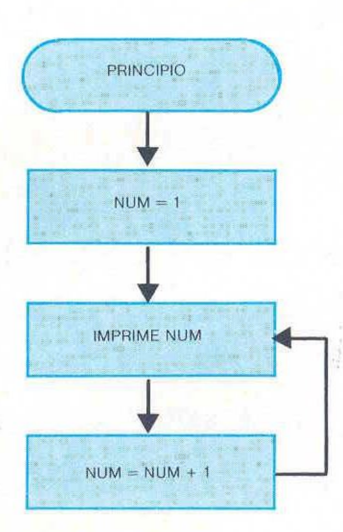
El contador se incrementa en 1.

```
40 GOTO 20
```

La línea 40 manda a la línea 20. ¿Pero, qué clase de igualdad es la de la línea 30?

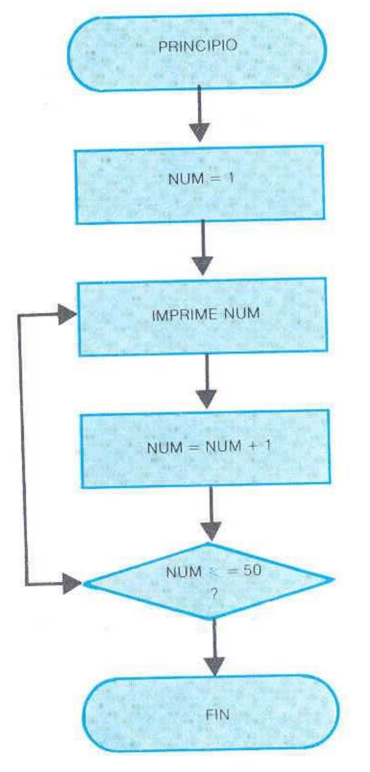
```
NUM = NUM + 1
```

Visto como una ecuación normal carece de sentido. Pero tu Spectrum no la interpreta de esta manera, sino como una signación: en primer lugar calcula la expresión a la derecha del igual y después pone su resultado en la variable NUM. Teclea estas instrucciones e indica el comando RUN. ¿Cómodo, verdad? Unicamente con cuatro intrucciones has resuelto el problema. Cuando te hayas cansado de ver pasar los números en pantalla, pulsa la tecla SPACE en cuanto veas el letrero **scroll?** En la base de la pantalla aparecerá el mensaje D BREAK-CONT REPEATS, 20 : 1.
Te habrás dado seguramente cuenta de que nuestro programa no satisface exactamente el objetivo que nos habíamos propuesto: continúa contando e imprimiendo los números mucho más allá de la meta de los 50.
La razón es bastante sencilla de localizar: representemos el programa en los términos de un diagrama de bloques:



PROGRAMACION

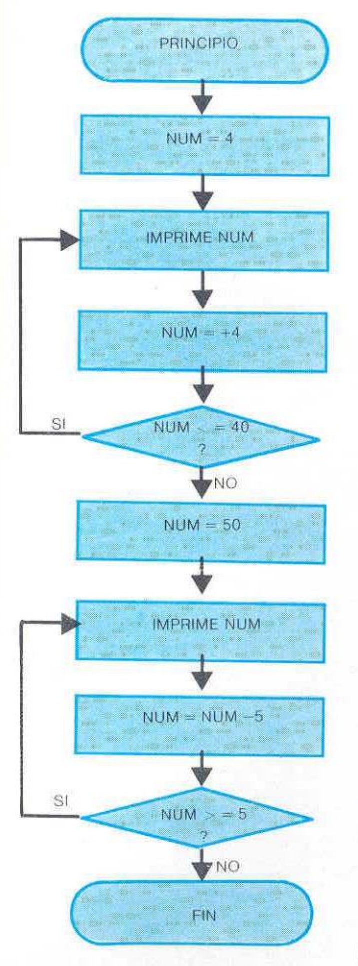
Tu Spectrum ejecutando el anterior programa entra (como se dice en la jerga informática) en un ciclo infinito, es decir, en una secuencia de instrucciones de la que no tiene forma de salir.
El error que hemos cometido es uno de los más frecuentes de aquellos en los que habitualmente incurre el programador primerizo: olvidarse de indicar una condición a satisfacer para que el programa pueda continuar.
El único remedio para este error lo constituye la inserción de un nuevo bloque, en cuyo interior se efectúe la comparación entre el contador y el límite máximo(en este caso igual a 50).
Por lo tanto la situación se convierte en:



A la que le corresponde el siguiente programa en BASIC:

```
10 LET NUM = 1
20 PRINT NUM
30 LET NUM = NUM + 1
40 IF NUM < = 50 THEN GOTO 20
50 FIN
```

La linea 40 constituye el «bloque de decisión». Naturalmente no es en absoluto obligatorio que el contador se incremente con un valor unitario; puede hasta ser decrementado (siempre que, naturalmente, la condición sea adecuada a la nueva circunstancia).
Lo que sigue es un ejemplo de tal situación; el objeto del programa es el de imprimir en orden creciente la tabla de multiplicar del 4, y en orden decreciente la del 5:



Al contador se le asigna, como valor de salida, el valor de 4

El incremento del contador es 4

Cuando el programa haya llegado a este punto la primera tabla (la del 4) ya habrá sido impresa.

Al contador se le asigna el valor de salida de la segunda tabla.

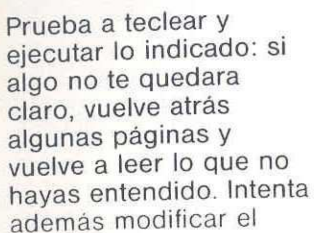
Esta vez el contador se decrementa.

En este punto la tabla del 5 en orden decreciente habrá sido impresa.

Las correspondientes instrucciones en BASIC son:

```
10 LET NUM = 4
20 PRINT NUM
30 LET NUM = NUM + 4
40 IF NUM < = 40 THEN GOTO 20
50 LET NUM = 50
60 PRINT NUM
70 LET NUM = NUM −5
80 IF NUM > = 5 THEN GOTO 60
```

PROGRAMACION



Prueba a teclear y ejecutar lo indicado: si algo no te quedara claro, vuelve atrás algunas páginas y vuelve a leer lo que no hayas entendido. Intenta además modificar el programa, para que imprima las tablas de multiplicar, por ejemplo, del 3 y del 7: este es un excelente sistema para hacer prácticas de programación. El contador es tambien un sistema para acordarnos de cuantas veces hemos realizado una operación.

Escribamos, por ejemplo, un programa que saque la media de un número indeterminado de valores, pero para hacer la media es necesario conocer exáctamente el

número de valores: ¿Cómo conseguirlo? Será suficiente con un sencillo contador que se incremente cada vez que introduzcamos un nuevo número:

```
10 CLS: PRINT "MEDIA DE VARIOS
   NUMEROS": PRINT
20 LET SUMA = 0 : LET NUM = 0
30 PRINT "INDICA FIN CON 0"
40 INPUT "NUMERO ="; N
50 IF N = 0 THEN IF NUM = 0  THEN GOTO 120
60 IF N = 0 THEN GOTO 80
70 LET SUMA = SUMA + N: LET NUM = NUM +
   1: GOTO 50
80 LET MEDIA = SUMA/NUM
90 PRINT "LA MEDIA DE"; NUM: "NUMEROS ES";
   MEDIA
100 INPUT "¿OTRA MEDIA? S/N"; R$
110 IF R$ = "S" THEN GOTO 20
120 REM FIN
```

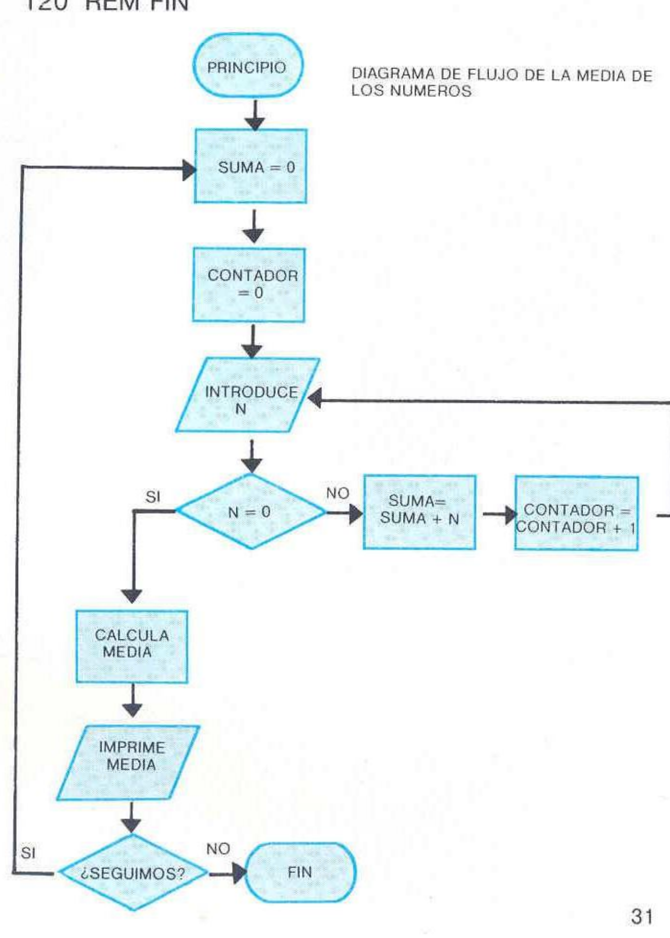


DIAGRAMA DE FLUJO DE LA MEDIA DE LOS NUMEROS

EJERCICIOS

Anota en el espacio en blanco el resultado que preveas para cada ejercicio, y verifica tu solución con el Spectrum. Aunque sólo cometas un único error, repasa la lección.

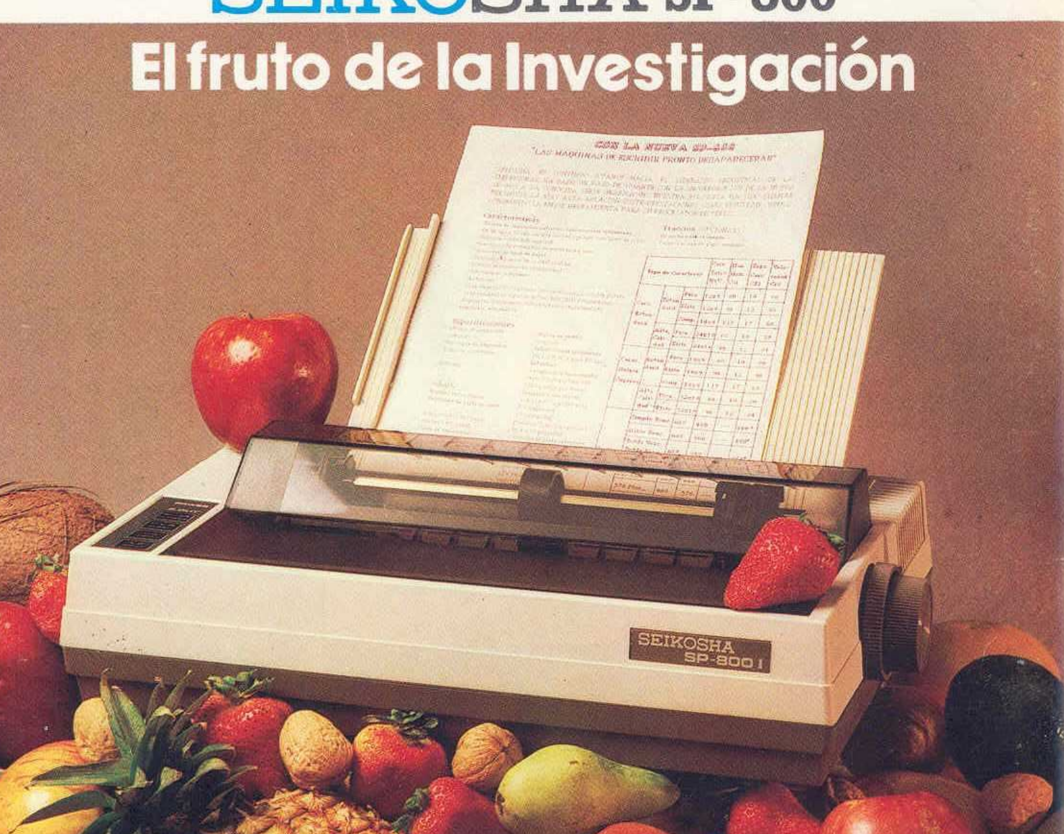
```
10 LET A = –416.4
20 LET B = INT (A)
30 PRINT B
40 IF A > B THEN PRINT "A > B"
50 IF A < B THEN PRINT "B > A"
```

```
10 LET A = 0 : LET B = .1: LET C = A – B
20 PRINT SGN (A)
30 PRINT SGN (B)
40 PRINT SGN (C)
```

```
10 S$ = "*" : C$ = " "
20 C$ = C$ + S$
30 PRINT C$
40 IF C$ = "* * * * * *" GOTO 60
50 GOTO 20
60 REM FIN
```

```
10 LET T = 100: LET C = 0: LET P = 20
20 PRINT C
30 LET C = C + P
40 IF C < = T THEN GOTO 20
50 REM FIN
```

SEIKOSHA SP-800



La nueva impresora de **SEIKOSHA SP-800**, con un ordenador personal puede escribir ***96 combinaciones de letra diferentes***, desde 96 caracteres por segundo a 20 con muy alta calidad de letra, además es gráfica en alta densidad.

Su precio es de 69.900 Pts con ***introductor automático hoja a hoja***.

Con un pequeño ordenador personal, un procesador de textos puede costar alrededor de cien mil pesetas.

Infórmese y comprenderá por qué **las máquinas de escribir** tienen demasiados años.

Nuestra calidad es "SEIKO";
nuestros precios, únicos
Si desea más información,
consulte con nuestro distribuidor
más cercano, llame o escriba a:

ESTOS SON NUESTROS MODELOS:

MODELO	VELOCIDAD	COLUMNAS	TIPOS DE LETRA	P.V.P.R.* INTERFACE PARALELO
GP-50 LA PEQUEñA	40 cps	46	2	25.900
GP-500 LA ECONOMICA	50 "	80	2	47.900
GP-550 LA STANDARD	80 "	80-136	10	59.900
SP-800 LA PERFECCION	96 "	80-137	20	69.900
GP-700 LA DE COLOR	50 "	80-106	3	84.900
BP-5200 LA DE OFICINA	200 "	136-272	10	199.900
BP-5420 LA MAS RAPIDA	420 "	136-272	10	299.900

* Los precios indicados son los recomendados para conexión tipo paralelo Centronics, para otro tipo de conexión, sufren un ligero incremento.

DiRAC

DIRECCION COMERCIAL:
Av. Blasco Ibañez, 114-116
46022 VALENCIA
Tel. (96) 372 88 89
Télex 62220
DIRECCION COMERCIAL EN CATALUñA:
C/Muntaner, 60-2-4Pta
08011 BARCELONA
Tel. (93) 323 32 19

Este pie de página ha sido realizado integramente con la nueva impresora:

SEIKOSHA SP-800